

Optimal Backpressure Scheduling in Wireless Networks using Mutual Information Accumulation

Jing Yang, *Member, IEEE*, Yanpei Liu, *Student Member, IEEE*, and Stark C. Draper, *Member, IEEE*

Abstract—In this paper we develop scheduling policies that maximize the stability region of a wireless network under the assumption that mutual information accumulation is implemented at the physical layer. When the link quality between nodes is not sufficiently high that a packet can be decoded within a single slot, the system can accumulate information across multiple slots, eventually decoding the packet. The result is an expanded stability region. The accumulation process over weak links is temporally coupled and therefore does not satisfy the independent and identically distributed (i.i.d) assumption that underlies many previous analysis in this area. Therefore the problem setting also poses new analytic challenges. We propose two dynamic scheduling algorithms to cope with the non-i.i.d nature of the decoding. The first performs scheduling every T slots, and approaches the boundary of the stability region as T gets large, but at the cost of increased average delay. The second introduces virtual queues for each link and constructs a virtual system wherein two virtual nodes are introduced for each link. The constructed virtual system is shown to have the same stability region as the original system. Through controlling the virtual queues in the constructed system, we avoid the non-i.i.d analysis difficulty and attain the full stability region. We derive performance bounds for both algorithms and compare them through simulation results.

Index Terms—Mutual information accumulation, backpressure algorithm, delay analysis, virtual queues.

I. INTRODUCTION

Because of the broadcast nature of wireless communications, multiple receiving nodes may overhear the transmission of a transmitting node in a wireless network. Therefore, a packet can be routed through different routes to its destination node. This “multiple-route diversity” can be exploited to improve various measure of network performance including throughput, average delay, and the probability of transmission failure.

Different routing and scheduling policies have been proposed to exploit the “broadcast advantage” of the wireless medium. Reference [1] proposes ExOR, a packet-level opportunistic routing protocol to exploit high loss-rate radio links in wireless networks. Reference [2] proposes a proactive link state routing protocol that improves the forward nodes selection in [1]. Reference [3] characterizes the maximum network capacity region when packet-level opportunistic routing is exploited, and proposes a routing algorithm named DIVBAR to stabilize the system. DIVBAR adopts the idea

of backpressure algorithms first proposed in [4], [5]. References [1]–[3] discuss how to select a forwarder based on the feedback information sent from the receivers that have successfully received the packets. With a similar maximum weight scheduling idea, [6] analyzes the optimal information theoretic based cooperative scheduling in a diamond network. The system is assumed to be able to adaptively select the optimal encoding-decoding scheme such that any desirable transmission rate vector within the corresponding capacity region can be achieved in every slot. This “fluid model” is not quite practical. However, because it is simple to analyze, it has been widely adopted in the literature.

The policy space explored in the above papers, in particular those having to do with wireless [1]–[3], [6], assumes that packet decoding is carried out within a single slot. This means that decoding depends only on current control actions and channel quality. However, there are many physical layer techniques, such as energy accumulation and mutual information accumulation (MIA), that do not fit this assumption. Such techniques do not require decoding to occur within a single slot but rather allow the receivers to accumulation observations across multiple time slots before decoding. This allows the network to exploit weak radio links more fully, thereby increasing system throughput.

In this paper we focus on mutual information accumulation and, while there is some prior research on resource optimization in networks using MIA [7]–[9], that work does not consider questions of network stability and protocol design. Our objectives in this work are therefore two fold. First, we want to characterize the maximum network capacity region when mutual information accumulation is implemented at the physical layer. Second, we aim to design joint routing and scheduling algorithms that stabilize any point within the capacity region.

To make the concept of mutual information accumulation more tangible, it helps to contrast it with energy accumulation. In energy accumulation multiple transmissions are combined non-coherently by receiving nodes. This is usually enabled by using space-time or repetition coding [10]–[12]. Mutual information accumulation is more efficient, the difference between the two being explained in [7]. Consider a pair of senders transmitting information over two independent additive white Gaussian noise channels to the same receiver. Assume transmitter power P and channel noise power N . Energy accumulation corresponds to the scenario when each transmitter sends the same codeword. When the decoder uses maximum ratio combining, a throughput of $\frac{1}{2} \log(1 + \frac{2P}{N})$ bits per channel use can be achieved. With mutual information

This work was supported in part by the Air Force Office of Scientific Research under grant FA9550-09-1-0140 and by the National Science Foundation under grant CCF 0963834.

J. Yang, Y. Liu and S. C. Draper are with the Department of Electrical and Computer Engineering, University of Wisconsin-Madison, WI, 53706, USA. e-mail: yangjing@ece.wisc.edu, yliu73@wisc.edu, sdraper@ece.wisc.edu.

accumulation, independent parity symbols are sent, and the system can achieve $2 \times \frac{1}{2} \log(1 + \frac{P}{N})$ bits per channel use, which outperforms energy accumulation. It has been noted in [7] that for Gaussian channels at low signal-to-noise ratio (SNR) energy accumulation is equivalent to mutual information accumulation because capacity at low SNR is linear in SNR. Mutual information accumulation can be realized through the use of rateless (or fountain) codes [7], [13], [14].

Similar to [1]–[3], we assume our system operates at packet-level, and each active transmitter transmits a packet in each slot. Different from the probabilistic channel model discussed in these references, we consider the scenario where the channel state varies slowly, so that link conditions can be assumed to be static over a long transmission period. Under this scenario, for a weak link whose link rate is below the transmission rate, a packet cannot be successfully delivered across the link in any single slot, even with repetitive attempts in different slots. Thus, the corresponding receiver can never decode and become a forwarder under the schemes discussed in [1]–[3]. However, when rateless codes are used by the transmitters, although the corresponding receiver of the weak link cannot decode the packet within a single slot, it can store that corrupted packet and accumulate more information in later slots. Eventually, a packet can be successfully delivered across a weak link after a number of transmissions by accumulating information across the slots. Thus, weak links can still be utilized in slowly changing network environments. Compared with opportunistic routing schemes, mutual information accumulation provides more reliable throughput over weak links, and doesn't require the feedback information to determine the forwarder in each slot. Thus it reduces the required overhead.

Compared to the networks in [6] and references [15], [16] where varying rate can be achieved within a slot through adaptive encoding-decoding schemes, the mutual information accumulation scheme doesn't require the encoding-decoding to be changed in every slot. However, on average, it can achieve the same rate by repetitive transmission and information bits accumulation. Therefore, the system is more practical to implement without sacrificing throughput. On the other hand, the information bit accumulation process also brings new challenges to the design and analysis of routing and scheduling algorithm.

The contribution of our work is three-fold.

- We characterize the maximum stability region of a wireless network enabled with mutual information accumulation under certain natural assumptions. Compared with networks where information accumulation is not allowed, the system is able to exploit weak links and an expanded stability region is thereby achieved.
- We propose two dynamic routing and scheduling policies to achieve the maximum stability region. Both policies require simple coordination and limited overhead.
- The techniques we develop to cope with the temporally coupled mutual information accumulation process are novel and may have use in queuing problems more widely.

The rest of the paper is organized as follows. In Section II, we describe the system model. In Section III, we present the

maximum stability region of the network under our system model. In Section IV and Section V, we design two different routing protocols to achieve the maximum stability region and analyze the performance. We present simulation evaluation in Section VII. Finally, we conclude in Section VIII. Proofs are deferred to the appendices.

II. SYSTEM MODEL

A. The Basic System Setting

We consider a time-slotted system with slots normalized to integral units $t \in \{0, 1, 2, \dots\}$. There are N network nodes, and links are labeled according to *ordered* node pairs (i, j) for $i, j \in \{1, \dots, N\}$. We assume that there are K different commodities in the network, $K \leq N$, where each commodity is labeled according to its destination node, e.g., all packets from commodity c should be routed to destination node c . Data arrives randomly in packetized units. Let $A_i^c(t)$ denote the number of packets from commodity c that exogenously arrive at network node i during slot t . Arrivals are assumed to be independent and identically distributed (i.i.d.) over timeslots, and we let $\lambda_i^c = E\{A_i^c(t)\}$ represent the arrival rate of commodity c into source node i in units of packets/slot. We assume $A_i^c(t) \leq A_{max}$ for all c, i, t .

We assume the channel fading states between any pair of nodes are stationary, and the active transmitters transmit with the same power level. Thus, a fixed reliable communication rate over an active link can be achieved in each slot. We expect that the algorithms developed in this paper can be modified to accommodate more general fading processes.

We use r_{ij} to denote the link rate between nodes i, j . We assume the link rate is reciprocal, i.e., $r_{ij} = r_{ji}$. We assume that each node can transmit at most one packet during any given node during any single time slot, i.e., $r_{ij} \leq 1$ packet/slot. We term the links with rate one packet per slot to be *strong* links; the rest of the links we term *weak* links. For weak links, we assume their rates are lower bounded by some constant value r_{min} , $0 < r_{min} < 1$. Define the set of neighbors of node i , $\mathcal{N}(i)$, as the set of nodes with $r_{ij} > 0$, $j \in \mathcal{N}(i)$. We assume the size of $\mathcal{N}(i)$, denoted as $|\mathcal{N}(i)|$, is upper bounded by a positive integer d . We define μ_{max} as the maximum number of packets a node can successfully decode in any slot, which is equivalent to the maximum number of nodes that can transmit to a single node simultaneously. Therefore, we have $\mu_{max} \leq d$.

We assume the system operates under constraints designed to reduce interference among transmitters. Under the interference model, in any timeslot, only a subset of nodes are allowed to transmit simultaneously. We assume that transmissions from nodes active at the same time are interference free. We denote the set of feasible *activation pattern* as \mathcal{S} , where an activation pattern $s \in \mathcal{S}$ represents a set of active nodes. With a little abuse of the notation, we interchangeably use s to represent an activation pattern and the set of active nodes of the pattern. For any $s \in \mathcal{S}$, we assume all of its subsets also belong to \mathcal{S} . This allows us to assert that all of the nodes in an activation pattern always transmit when that activation pattern is selected. This interference model can accommodate networks with orthogonal channels, restricted TDMA, etc.

B. Mutual Information Accumulation at Physical Layer

We assume mutual information accumulation is adopted at physical layer. Specifically, we assume that if a *weak* link with $r_{ij} < 1$ is activated, rateless codes are deployed at its corresponding transmitter. When a packet transmitted over a weak link cannot be successfully decoded during one slot, instead of discarding the observations of the undecodable message, the receiver stores that partial information, and accumulates more information when the same packet is retransmitted. A packet can be successfully decoded when the accumulated mutual information exceeds the packet size. The assumption $r_{ij} > r_{min}$ implies that for any active link, it takes at most some finite number of time slots, $\lceil 1/r_{min} \rceil$, to decode a packet.

In order to simplify the analysis, we assume that $1/r_{ij}$ is an integer. The reason for this choice will become clear when our algorithm is proposed in Section III. If r_{ij} does not satisfy this assumption, we round it down to $1/\lceil 1/r_{ij} \rceil$ to enforce it.

The challenge in extending the backpressure routing framework to systems that use mutual information accumulation is illustrated through the following example. Suppose that a node i has already accumulated half of the bits in packet 1 and half of the bits in packet 2. Since neither of the packets can be decoded, none of these bits can be transferred to the next hop, even though the total number of bits at node i is equal to that of a full packet. This means that we need to handle the undecoded bits in a different manner. We also observe that, if node i never accumulates enough information for packets 1 or 2, then these packets can never be decoded and will be stuck at node i . If we assume that the system can smartly drop the undecoded partial packet whenever a fully decoded copy is successfully delivered to its destination, coordination among the nodes is required to perform this action. The overhead of the coordination might offset the benefits brought by mutual information accumulation. Moreover, unlike the opportunistic model studied in [1]–[3], given that weak link is active, whether or not a successful transmission will occur in that slot is not an i.i.d random variable but rather a deterministic function of the number of already accumulated bits of that packet at the receiving node. This difference makes the analysis even more complicated.

Therefore, in the following sections, we define two different types of queues. One is the traditional *full packet* queue which stores fully decoded packets at each node. The other type of queue is a *partial packet* queue. It represents the fraction of accumulated information bits from a particular packet. The specific definition of the queues and their evolution depends on the scheduling policy, and will be clarified in Section IV, and Section V, respectively.

We assume each node has infinite buffer space to store fully decoded packets and partial packets. Overflow therefore does not occur.

C. Reduced Policy Space

The policy space of the network we consider can be much larger than that of networks without mutual information accumulation. First, because of the broadcast nature of wireless

communication, multiple receivers can accumulate information regarding the same packet in any given slot, and a receiver can collect information on a single packet from multiple nodes. Keeping track of the locations of different copies of the same packet requires a lot of overhead. Second, allowing multiple receivers to store copies of the same packet introduces more traffic into the network. Therefore, stabilizing the network requires a sophisticated centralized control strategy. Finally, accumulating information bits of a packet from multiple nodes makes the decoding options of a packet increase exponentially; thus characterizing the network maximum stability region becomes intractable. Therefore, we make the following assumptions:

- A1. *For each packet in the system, at any given time, only one node is allowed to keep a fully decoded copy of it.*
- A2. *In addition to the node with the fully decoded copy, one other node is chosen as the potential forwarder for the packet. Only the potential forwarder is allowed to accumulate information about the packet.*

Restricting ourselves to this policy space may sacrifice some part of the stability region that could be achieved under a more general policy space. However, as we will see, these assumptions enable us to explicitly characterize the maximum stability region with the given policy space. Compared with systems that operate without the capability to accumulate mutual information, our system is able to exploit weak links even when one-slot decoding is not possible. The stability region will thereby be greatly enlarged.

If a node directly contributes to the successful decoding of a packet at another node, this node is denoted as a *parent* for that packet. Assumptions A1-A2 guarantee that for any packet in the network, there is only one parent at any given time. We also note that, if we relax assumptions A1-A2, and make the following assumption instead

- A3. *Every packet existing in the network has a single parent at any given time. In other words, the accumulated information required to decode a packet at a node is received from a single transmitting node.*

then, the maximum stability region under A1-A2 and that under A3 are equivalent. Assumption A3 means that we don't allow a node to accumulate information from different nodes (different copies of the same packet) to decode that packet. However, multiple copies of a packet may still exist in the network.

III. NETWORK CAPACITY WITH MUTUAL INFORMATION ACCUMULATION

In this section we characterize the optimal throughput region under all possible routing and scheduling algorithms that conform to the network structure specified in Section II-A and Assumption A3.

At the beginning of each slot, a certain subset s of nodes is selected to transmit in the slot. Any node $i \in s$ can transmit any packet that it has received *and* successfully decoded in the past. Note that because packets must be decoded prior to transmission, partial packets cannot be delivered. For node $j \in \mathcal{N}(i)$, if it has already decoded a packet being transmitted,

it can simply ignore the transmission; otherwise, it listens to the transmission and aims to decode it at the end of that slot. Receiving nodes connected with strong links can decode the packet in one slot. Nodes connected with weak links cannot decode the packet in one slot. Rather, they need to listen to the same node transmitting the same packet over a number of slots. A packet is said to be successfully delivered to its destination node when the *first copy* of the packet is decoded by its destination node. These assumptions allow for any possible routing and scheduling policy satisfying Assumption A3. We note that packets arrive at a node from a single commodity may get delivered to their destination node in a permuted order since each packet may take a different route.

Let λ represent the input rate vector to the system, where λ_i^c is the input rate of commodity c entering node i . Define $Y_i^c(t)$ as the number of packets from commodity c that originated at node i and have been successfully delivered to destination node c over $[0, t)$. According to the definition of network stability [17], a policy is defined as *stable* if

$$\lim_{t \rightarrow \infty} \frac{Y_i^c(t)}{t} = \lambda_i^c, \quad \forall c. \quad (1)$$

Stronger definitions of stability can be found in [4], [5], [18]. The maximum stability region or *network layer capacity region* Λ of a wireless network with mutual information accumulation is defined as the closure of all λ that can be stabilized by the network according to some policy with the structure described above.

Theorem 1 *For a network with given link rates $\{r_{ij}\}$ and a feasible activation pattern set \mathcal{S} , the network capacity region Λ under assumption A3 consists of all rate vectors $\{\lambda_n^c\}$ for which there exists flow variables $\{\mu_{ij}^c\}$ together with a probability π_s for each possible activation pattern $s \in \mathcal{S}$ such that*

$$\mu_{ij}^c \geq 0, \quad \mu_{ci}^c = 0, \quad \mu_{ii}^c = 0, \quad \forall i, j, c \quad (2)$$

$$\sum_l \mu_{li}^c + \lambda_i^c \leq \sum_j \mu_{ij}^c, \quad \forall i \neq c, \forall c \quad (3)$$

$$\sum_c \mu_{ij}^c \leq \sum_c \sum_{s \in \mathcal{S}} \pi_s \theta_{ij}^c(s) r_{ij}, \quad \forall i, j \quad (4)$$

$$\sum_{s \in \mathcal{S}} \pi_s \leq 1, \quad (5)$$

where the probabilities $\theta_{ij}(s)$ satisfy

$$\theta_{ij}^c(s) = 0 \text{ if } i \notin s, \quad (6)$$

$$\sum_c \sum_j \theta_{ij}^c(s) = 1, \forall i. \quad (7)$$

The necessity of this theorem can be proved following the same approach in [3] and is provided in Appendix A. The sufficiency part is proved in Section IV by constructing a stabilizing policy for any rate vector λ that is in the interior of capacity region.

The capacity region is essentially similar to the capacity theorem of [3], [17]. The relations in (2) represent non-negativity and flow efficiency constraints for conservation constraints. Those in (3) represent flow conservation constraints.

Those in (4) represent link constraints for each link (i, j) . The variable $\theta_{ij}^c(s)$ can be interpreted as the probability that the transmission over link (i, j) eventually contributes to the delivery of a packet of commodity c at node c , given that the system operates in pattern s . In other words, link (i, j) is on the routing path for this packet from its origin node to its destination node.

This theorem implies that the network stability region under Assumption A3 can be defined in terms of an optimization over the class of all stationary policies that use only single-copy routing. Thus, for any rate vector $\lambda \in \Lambda$, there exists a stationary algorithm that can support that input rate vector by single-copy routing all data to the destination.

The Λ defined above are in the same form as when a “fluid model” is considered. In other words, the extra decoding constraint imposed by mutual information accumulation does not sacrifice any part of the stability region. We can simply ignore the packetization effect when we search for the maximum stability region.

The $\sum_c \mu_{ij}^c$ defining the stability region represents the *effective* flow rate over link (i, j) . An “effective” transmission means that the bits transferred by that transmission eventually get delivered to the destination. If the transferred information becomes a redundant copy, or discarded partial packet, the transmission is not effective, and doesn’t contribute to the effective flow rate. We can always make the inequalities (3)-(4) tight by controlling the value of π_s .

Solving for the parameters $\{\pi_s\}$ and $\{\theta_{ij}^c(s)\}$ required to satisfy the constraints requires a complete knowledge about the set of arrival rates $\{\lambda_i^c\}$, which cannot be accurately measured or estimated in real networks. On the other hand, even when λ is given, solving the equations can still be quite difficult. In the following, we overcome this difficulty with online algorithms which stabilize any λ within Λ , but with a possibly increased average delay as λ approaches the boundary of Λ .

IV. T-SLOT DYNAMIC CONTROL ALGORITHM

In the following, we construct a policy that fits Assumptions A1-A2. Although more restrictive than Assumption A3, we will see that the stronger assumptions do not compromise stability performance in the sense that they do not reduce the stability region. To construct a dynamic policy that stabilizes the system anywhere in the interior of Λ , specified in Theorem 1, we first define our decision and queues variables.

We assume that each packet entering the system is labeled with a unique index k . At time t , $0 \leq k \leq \sum_{c,i} \sum_{\tau=1}^t A_i^c(\tau)$. If packet k belongs to commodity c , we denote it as $k \in \mathcal{T}_c$. Let $\{\beta_{ij}^{(k)}(t)\}$ represent the binary control action of the system at time t . Specifically, $\beta_{ij}^{(k)}(t) = 1$ means that at time t , node i transmits packet k to node j . We restrict the possible actions so that in each slot each node transmits at most one packet, i.e.,

$$\sum_{j,k} \beta_{ij}^{(k)}(t) \leq 1, \quad \forall i, \quad (8)$$

and at most one node is chosen as the forwarder for packet k ,

i.e.,

$$\sum_{i,j} \beta_{ij}^{(k)}(t) \leq 1, \quad \forall k. \quad (9)$$

Because of the mutual information accumulation property, even if packet k is transmitted over link (i, j) in slot t , it doesn't necessarily mean that packet k can be decoded at node j at the end of slot t . In particular, under the fixed link rate assumption, successful transmission cannot occur over weak links in a single timeslot. We let $f_{ij}^{(k)}(t)$ be an indicator function where $f_{ij}^{(k)}(t) = 1$ indicates that the packet k has been successfully delivered from node i to node j in slot t . The indicator function is a function of the current control action and partial queue status at the beginning of slot t . Apparently, $f_{ij}^{(k)}(t) = 1$ implies that $\beta_{ij}^{(k)}(t) = 1$.

As discussed in Section II-B, we define two types of queues at each nodes. One is to store the fully received and successfully decoded packet at the nodes, while the other queue stores the partially received packets. We use $Q_i^c(t)$ to denote the length of node i 's queue of fully received packets from commodity c at time t , and use $P_i^{(k)}(t)$ to represent the total fraction of packet k accumulated by node i up to time t . The sum-length of partial queues of commodity c at node i storing partial packets can be represented as $P_i^c(t) = \sum_{k \in \mathcal{T}_c} P_i^{(k)}(t)$. The fraction of packet k , $P_i^{(k)}(t)$, can be cleared either when packet k is successfully decoded and enters the full packet queue Q_i^c , or when the system controller asks node i to drop packet k . With a little abuse of notation, we use Q_i^c and P_i^c to denote the full packet queue and partial packet queue from commodity c at node i , respectively.

Then, according to our Assumptions A1-A2, the queue lengths evolve according to

$$Q_i^c(t+1) = \left(Q_i^c(t) - \sum_{j,k \in \mathcal{T}_c} \beta_{ij}^{(k)}(t) f_{ij}^{(k)}(t) \right)^+ + \sum_{l,k \in \mathcal{T}_c} \beta_{li}^{(k)}(t) f_{li}^{(k)}(t) + A_i^c(t) \quad (10)$$

$$P_i^{(k)}(t+1) = P_i^{(k)}(t) + \sum_l \beta_{li}^{(k)}(t) r_{li}^{(k)}(t) - \sum_l \beta_{li}^{(k)}(t) f_{li}^{(k)}(t) - \sum_{l, (m \neq i)} P_i^{(k)}(t) \beta_{lm}^{(k)}(t) \quad (11)$$

where

$$r_{li}^{(k)}(t) = \begin{cases} r_{li} & P_i^{(k)}(t) + r_{li} \leq 1 \\ 1 - P_i^{(k)}(t) & P_i^{(k)}(t) + r_{li} > 1 \end{cases} \quad (12)$$

and $(x)^+ = \max\{x, 0\}$. Under the assumption that $1/r_{ij}$ is an integer for every (i, j) , we have $r_{li}^{(k)}(t) = r_{li}$.

Since we only allow there to be only a forwarder for any given packet at any time, if $\beta_{lm}^{(k)}(t) = 1$, any nodes other than node m which have accumulated partial information of packet k must drop that partial packet k . This effect results in the last negative term in (11). On the other hand, the first negative term in (11) corresponds to successful decoding of packet k , after which it is removed and enters Q_i^c for some c .

A. The T -slot Algorithm

Our algorithm works across epochs, each consisting of T consecutive timeslots. Action decisions are made at the start of each epoch and hold constant through the epoch. We analyze the choice of T on the stability region and average backlog. Any rate vector λ inside the capacity region Λ can be stabilized by a sufficiently large choice of T .

- 1) **Check single-link backpressure.** At the beginning of each epoch, i.e., when $t = 0, T, 2T, \dots$, node i checks its neighbors and computes the differential backlog weights

$$W_{ij}(t) = \max_c [Q_i^c(t) - Q_j^c(t)]^+ r_{ij}, \quad j \in \mathcal{N}(i).$$

Denote the maximizing commodity as

$$c_{ij}^* = \arg \max_c [Q_i^c(t) - Q_j^c(t)]^+.$$

- 2) **Select forwarder.** Choose the potential forwarder for the packets in Q_i as the node j with the maximum weight $W_{ij}(t)$. Denote this node as $j_i^* = \arg \max_j W_{ij}(t)$.
- 3) **Choose activation pattern.** Define the activation pattern s^* as the pattern $s \in S$ that maximizes

$$\sum_{i \in s} W_{ij_i^*}.$$

Any node $i \in s^*$ with $W_{ij_i^*} > 0$ transmits packets of commodity $c_{ij_i^*}^*$ to j_i^* . The pairing of transmitter $i \in s^*$ and receiver j_i^* and the commodity being transmitted $c_{ij_i^*}^*$ is continued for T consecutive timeslots.

- 4) **Clear partial queues.** Release all the accumulated bits in the partial queue P_i^c , $\forall i, c$, at the end of each epoch.

The T -slot algorithm satisfies constraints (8)-(9). The "potential forwarder" in Step 2) refers to the forwarder of node i if node i is active. We clear all of the partial queues in the system every T slots (in Step 4)) for the simplicity of analysis. It is likely not the best approach to handle the partial queues. Intuitively, the performance should be improved if we only release the partial queues when a selected forwarder for a packet is not the same as the previous one (thus satisfying A3).

Theorem 2 *The algorithm stabilizes any rate vector satisfying $\lambda + \epsilon(T) \in \Lambda$, where $\epsilon(T)$ is a vector with minimum entry $\epsilon > 1/T$. The average expected queue backlog $\lim_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{c,i} \mathbb{E}\{Q_i^c(\tau)\}$ in the system is upper bounded by*

$$\frac{KNT^2(\mu_{max} + A_{max})^2 + NT^2}{2(\epsilon T - 1)} + \frac{KN(T-1)(\mu_{max} + A_{max})}{2}.$$

The proof of Theorem 2 is provided in Appendix B. The proof is based on the fact that the T -slot algorithm minimizes the T -slot Lyapunov drift, which is shown to be negative when $\sum_{c,i} Q_i^c(t)$ is sufficiently large.

The constructed algorithm proves the sufficiency of Theorem 1. The intuition behind the algorithm is that by using the weak links consecutively over a long window, the potentially contributing loss caused by dropped partial packets is kept small, therefore, the effective rates over the weak links can get

close to the link rate. The algorithm approaches the boundary of the capacity region in $O(1/T)$.

When T is large enough, the average expected backlog in the system scales as $O(T)$. For this reason, in the next section, we introduce a virtual queue based algorithm which updates action every single slot. We expect that average backlog under the virtual queue based algorithm will be improved since its upper bound does not scale as $O(T)$.

Given $T > \frac{1}{\epsilon}$, the upper bound in Theorem 2 is a convex function of T . This implies that for any $\lambda + \epsilon(T) \in \Lambda$, there exists an optimal value of T which stabilizes the system and introduces minimal delay bound. However, when the arrival rates are unknown, it may not be practical to search for this optimal value.

Finally, we note that for some special values of T , the network can still be stabilized even when $T \leq 1/\epsilon$. For example, when T is chosen as $\prod_{(i,j)} \frac{1}{r_{ij}}$, then, under any possible activation pattern $s \in \mathcal{S}$, all partial packets are decoded at the end of the T -slot window. This implies that the policy can stabilize any $\lambda + \epsilon \in \Lambda$. This phenomena will be illustrated through examples in Section VII. For small networks, such a value for T can be easily computed and may be small; for large networks with many weak links, such value may still be quite large.

V. VIRTUAL QUEUE BASED ALGORITHM

In this section, we develop a second algorithm that exhausts the stability region without needing to set a large T . Thereby it attains better delay performance. As seen in Theorem 1, the delay is caused by the long time window of planning and infrequent update of the control action. Therefore, in order to obtain better delay performance, intuitively, we need to update our policy more frequently. This requires us to design more sophisticated mechanisms to handle the partial packet queues and additional analysis challenges brought by the temporally coupled decoding process over weak links.

Our approach is to construct a network that contains “virtual” queues, which handle the partial packets and decoding process over weak links. The resulting network has the same maximum stability region as the original network. By stabilizing the constructed network, the original network is also stabilized.

Specifically, in order to handle the partial packet queue in a simple and effective way, we introduce buffers over weak links. We assume there is a buffer at the transmitter side for each weak link. Then, if a node wants to send a packet over a weak link, the packet is pushed into the buffer. The buffer keeps the packet until it is successfully decoded at the corresponding receiver. The intuition behind the introduction of these buffers is that, since we don’t want dropped partial packets to lose much effective rate over weak links, once the system picks a forwarding node for a particular packet, the system never changes this decision.

For transmissions over weak links, a packet can only be decoded and transferred to next hop when enough information is collected. Under the T -slot algorithm, since control actions updates every T slot, and partial queues are cleared at the end

of every epoch, it is relatively simple to track the queue evolution and perform the analysis. When control actions change every slot, queues may evolve in a more complicated way and thus difficult to track and analyze. In order to overcome the analysis challenges, we introduce a second buffer at the receiver side of each weak link. Under the proposed algorithm, we ensure that the receiver never tries to decode a packet until it accumulates enough information, i.e., queue length in the second buffer only decreases when it is sufficiently long. By doing this, the evolution of the queue lengths can be tracked and analyzed easily.

Essentially, we only need to introduce virtual nodes and virtual buffers over weak links in order to handle partial packets. However, link rates may vary over time and vary in different activation patterns (discussed in Sec. VI-B). Therefore, for the virtual queue based algorithm, we introduce virtual nodes and buffers over both weak links and strong links, and treat them uniformly.

A. The Virtual Queue Vector

We divide $Q_i^c(t)$ into two parts. The first stores the packets that have not yet been transmitted in any previous slots, denoted as $U_i^c(t)$. The second stores packets partially transmitted over some links but not yet decoded, denoted as $V_i^c(t)$. Since each packet in the second part is associated with some link, in order to prevent any loss of effective rate caused by dropped partial packets, we require these packets to be transmitted over the same link until they are decoded. We use $V_{ij}^{(k)}(t)$ to denote the information of packet k required to be transmitted over link (i, j) , and $P_{ij}^{(k)}(t)$ to denote the accumulated information of packet k at node j . We define $V_{ij}^c(t) = \sum_{k \in \mathcal{T}_c} V_{ij}^{(k)}(t)$, and $P_{ij}^c(t) = \sum_{k \in \mathcal{T}_c} P_{ij}^{(k)}(t)$, where we recall that \mathcal{T}_c is the set of packets of commodity c . Note that $P_{ij}^c(t)$ is different from $P_j^c(t)$ defined in Section IV, since the latter is associated with node j and the former is associated with link (i, j) .

Associated with virtual queues, we define *virtual nodes*, as depicted in Fig. 1. For the link from node i to node j , we associate one virtual node with $\{V_{ij}^c\}_c$ and a second with $\{P_{ij}^c\}_c$. The virtual node associated with $\{V_{ij}^c\}_c$ is denoted as v_{ij} , while the virtual node associated with $\{P_{ij}^c\}_c$ is denoted as p_{ij} . We have decomposed the weak link (i, j) into three links: (i, v_{ij}) , (v_{ij}, p_{ij}) , (p_{ij}, j) , with link rates 1, r_{ij} , 1, respectively. The virtual nodes and corresponding link rates for link (j, i) can be defined in a symmetric way.

We follow the definition of control actions, where $\{\beta_{ij}^{(k)}(t)\}$ represent the control action of the system at time t . Depending on whether or not packet k has already been transmitted by node i , we also divide the decision actions into two types, denoted as $\beta_{ij}^{1(k)}$ and $\beta_{ij}^{2(k)}$, respectively. In the following algorithm, we only make control decisions for the packets at the head of corresponding queues. Therefore we can replace the superscript packet index (k) by its commodity c without any worry of confusion.

When $\beta_{ij}^{1c}(t) = 1$, node i pushes a *new* packet from $U_i^c(t)$ into the tail of V_{ij}^c at the beginning of slot t . This implies that the system assigns node j to be the next forwarder for

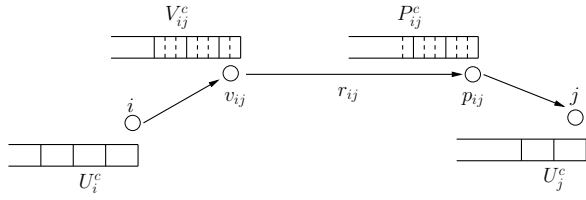


Fig. 1. The constructed virtual system.

that packet. Once the packet is pushed into V_{ij}^c , we transmit the packet that is at the head of V_{ij}^c to node j , generally a different packet. Thus, an amount r_{ij} of information can be accumulated at the tail of $P_{ij}^c(t)$, and the length of V_{ij}^c is reduced by r_{ij} . This mechanism ensures that the packets in the virtual buffer is transmitted and decoded in a FIFO fashion.

When $\beta_{ij}^{2c}(t) = 1$, without pushing a new packet into the buffer, we retransmit the packet at the head of $V_{ij}^c(t)$. We let

$$\beta_{ij}^c(t) = \beta_{ij}^{1c}(t) + \beta_{ij}^{2c}(t). \quad (13)$$

We require that

$$\sum_{c,j} \beta_{ij}^c(t) \leq 1, \quad \forall i, t. \quad (14)$$

Further, we define $f_{ij}^c(t) \in \{0, 1\}$ as binary decoding control actions. $f_{ij}^c(t) = 1$ indicates that receiver j has accumulated enough information to decode the packet at the head of $P_{ij}^c(t)$. It then moves that packet out of $P_{ij}^c(t)$ and into $U_j^c(t)$. We impose the following constraint

$$f_{ij}^c(t) \leq P_{ij}^c(t), \quad \forall c, i, j \quad (15)$$

which indicates that $f_{ij}^c(t) = 1$ only when $P_{ij}^c(t) \geq 1$, i.e., receiver j has accumulated enough information to decode the packet at the head of $P_{ij}^c(t)$. The reason for imposing this constraint on $f_{ij}^c(t)$ is that, we cannot simply use $(P_{ij}^c(t) - f_{ij}^c(t))^+$ to represent the queue length of P_{ij}^c after a decoding action is taken at that queue. If $P_{ij}^c(t) < 1$, even if $f_{ij}^c(t) = 1$, after the decoding action, the queue length will still be $P_{ij}^c(t)$ since no packet can be successfully decoded. This is not equal to $(P_{ij}^c(t) - f_{ij}^c(t))^+$, which is zero in this scenario.

Then, according to constraints (14) and (15), the queue lengths evolve according to

$$U_i^c(t+1) = \left(U_i^c(t) - \sum_j \beta_{ij}^{1c}(t) \right)^+ + \sum_l f_{li}^c(t) + A_i^c(t) \quad (16)$$

$$\begin{aligned} V_{ij}^c(t+1) &\leq (V_{ij}^c(t) + \beta_{ij}^{1c}(t)(1 - r_{ij}) - \beta_{ij}^{2c}(t)r_{ij})^+ \\ &= (V_{ij}^c(t) + \beta_{ij}^{1c}(t) - \beta_{ij}^c(t)r_{ij})^+ \end{aligned} \quad (17)$$

$$P_{ij}^c(t+1) \leq P_{ij}^c(t) + \beta_{ij}^c(t)r_{ij} - f_{ij}^c(t) \quad (18)$$

The inequalities in (17) and (18) come from the fact that $\beta_{ij}^{1c}(t)$ and $\beta_{ij}^{2c}(t)$ can be applied to a dummy packet when a queue is empty. When the corresponding queue is not empty, the inequality becomes an equality.

Define

$$\begin{aligned} L(\mathbf{U}(t), \mathbf{V}(t), \mathbf{P}(t)) &= \sum_{c,i} (U_i^c(t))^2 + \sum_{c,(i,j)} (V_{ij}^c(t))^2 \\ &\quad + \sum_{c,(i,j)} (P_{ij}^c(t) - \eta)^2 \end{aligned} \quad (19)$$

where η is a parameter used to control the length of $P_{ij}^c(t)$. Define $\Delta(t)$ as the one-slot sample path Lyapunov drift:

$$\Delta(t) := L(\mathbf{U}(t+1), \mathbf{V}(t+1), \mathbf{P}(t+1)) - L(\mathbf{U}(t), \mathbf{V}(t), \mathbf{P}(t))$$

Lemma 1 Under constraints (14) and (15), the sample path Lyapunov drift satisfies

$$\begin{aligned} \Delta(t) &\leq 2 \sum_{c,i} U_i^c(t) A_i^c(t) - 2 \sum_{c,i,j} [U_i^c(t) - V_{ij}^c(t)] \beta_{ij}^{1c}(t) \\ &\quad - 2 \sum_{c,i,j} [V_{ij}^c(t) - P_{ij}^c(t)] r_{ij} \beta_{ij}^c(t) \\ &\quad - 2 \sum_{c,i,j} [P_{ij}^c(t) - \eta - U_j^c(t)] f_{ij}^c(t) + \alpha_2 \end{aligned} \quad (20)$$

where

$$\alpha_2 = KN(d + A_{max})^2 + 2N + KNd \quad (21)$$

The proof of this lemma is provided in Appendix C.

B. The Algorithm

In contrast to the algorithm of Section IV, this algorithm updates every timeslot. The purpose of the algorithm is to minimize the right hand side of (20) given the current $\mathbf{U}, \mathbf{V}, \mathbf{P}$.

- 1) **Find per-link backpressure.** At the beginning of a timeslot, node i checks its neighbors and computes the differential backlogs. We compute the weight for the link between node i and the first virtual node, and, separately, the weight for the link between the two virtual nodes. Specially, the weight for control action β_{ij}^{1c} is computed as

$$W_{ij}^{1c}(t) = [U_i^c(t) - V_{ij}^c(t) + (V_{ij}^c(t) - P_{ij}^c(t))r_{ij}]^+$$

and the weight for control action β_{ij}^{2c} is computed as

$$W_{ij}^{2c}(t) = [V_{ij}^c(t) - P_{ij}^c(t)]^+ r_{ij}$$

The weight of commodity c over link (i, j) is $W_{ij}^c(t) = \max\{W_{ij}^{1c}(t), W_{ij}^{2c}(t)\}$. The weight for the link (i, j) is

$$W_{ij}(t) = \max_c W_{ij}^c(t),$$

and the optimal commodity

$$c_{ij}^* = \arg \max_c W_{ij}^c(t).$$

- 2) **Select forwarder.** Choose the potential forwarder of the current slot for node i with the maximum weight $W_{ij}(t)$ and denote it as $j_i^* = \arg \max_j W_{ij}(t)$.
- 3) **Choose activation pattern.** Define the optimal activation pattern s^* as the pattern $s \in S$ that maximizes

$$\sum_{i \in S} W_{ij_i^*}.$$

- 4) **Transmit packets.** For each $i \in s^*$, if $W_{ij_i^*} > 0$, let node i transmit a packet of commodity $c_{ij_i^*}^*$ to node j_i^* . For strong links, node i transmits a packet from the head of $U_i^{c^*}$. If link (i, j^*) is a weak link, and $W_{ij}(t) = W_{ij}^{1c}(t)$, node i pushes a new packet from $U_i^{c^*}$ into $V_{ij_i^*}^{c^*}$ and

transmits the packet from the head of $V_{ij}^{c*}(t)$; otherwise, node i resends the packet at the head of $V_{ij}^{c*}(t)$.

- 5) **Decide on decoding actions.** For each link (i, j) and each commodity c , we choose $f_{ij}^c(t) \in \{0, 1\}$ to maximize

$$[P_{ij}^c(t) - \eta - U_j^c(t)]f_{ij}^c(t) \quad (22)$$

where η is a parameter greater than or equal to 1. We let $f_{ij}^c(t) = 1$ when $P_{ij}^c(t) - \eta - U_j^c(t) = 0$.

Lemma 2 *Under the above virtual queue based algorithm: (a) If $P_{ij}^c(t) < \eta$ for some weak link (i, j) and slot t , then $f_{ij}^c(t) = 0$. (b) If $P_{ij}^c(t_0) \geq \eta - 1$, then, under the proposed algorithm, $P_{ij}^c(t) \geq \eta - 1$ for every $t \geq t_0$.*

Proof: In order to maximize $(P_{ij}^c(t) - \eta - U_j^c(t))f_{ij}^c(t)$, $f_{ij}^c(t) = 1$ only when $P_{ij}^c(t) - \eta - U_j^c(t) > 0$. Therefore, if $P_{ij}^c(t) < \eta$, $f_{ij}^c(t)$ must equal zero, which proves (a).

Now suppose that $P_{ij}^c(t) \geq \eta - 1$ for some slot t . We show that it also holds for $t + 1$. If $P_{ij}^c(t) \geq \eta$, then it can decrease by at most one packet on a single slot, so that $P_{ij}^c(t + 1) \geq P_{ij}^c(t) - f_{ij}^c(t) \geq \eta - 1$. If $P_{ij}^c(t) < \eta$, we must have $f_{ij}^c(t) = 0$, the queue cannot decrease in slot t , and we again have $P_{ij}^c(t + 1) \geq \eta - 1$. ■

With Lemma 2, we can see that when setting $\eta = 1$, under the proposed algorithm, if $P_{ij}^c(t) < 1$ for some weak link (i, j) and slot t , then $f_{ij}^c(t) = 0$. $f_{ij}^c(t)$ can only equal one when $P_{ij}^c(t) \geq 1$. Thus, constraint (15) is satisfied automatically for every slot under the proposed algorithm.

Theorem 3 *For a network with given link rates $\{r_{ij}\}$ and a feasible activation pattern set S , the network capacity region Λ' for the constructed network consists of all rate matrices $\{\lambda_n^c\}$ for which there exists flow variables $\{\mu_{ij}^{vc}\}$, $v = 1, 2, 3$ together with probabilities π_s for all possible activation pattern $s \in S$ such that*

$$\mu_{ij}^{vc} \geq 0, \quad \mu_{ci}^{vc} = 0, \quad \mu_{ii}^{vc} = 0, \quad \forall i, j, v, c \quad (23)$$

$$\sum_l \mu_{li}^{3c} + \lambda_i^c \leq \sum_j \mu_{ij}^{1c}, \quad \forall i \neq c \quad (24)$$

$$\mu_{ij}^{1c} \leq \mu_{ij}^{2c}, \quad \mu_{ij}^{2c} \leq \mu_{ij}^{3c} \quad \forall i, j, c \quad (25)$$

$$\sum_c \mu_{ij}^{2c} \leq \sum_c \sum_{s \in S} \pi_s \theta_{ij}^c(s) r_{ij}, \quad \forall i, j \quad (26)$$

$$\sum_{s \in S} \pi_s \leq 1 \quad (27)$$

where the probabilities $\theta_{ij}^c(s)$ satisfies

$$\theta_{ij}^c(s) = 0 \text{ if } i \notin s, \quad \sum_{c,j} \theta_{ij}^c(s) = 1, \forall i \quad (28)$$

Proof: The necessary part can be proved in the same way for Theorem 1. The sufficiency will be proved through constructing an algorithm that stabilizes all rate vectors satisfying the constraints. ■

In this constructed virtual network, $\mu_{ij}^{1c}, \mu_{ij}^{2c}, \mu_{ij}^{3c}$ can be interpreted as the flow over links (i, v_{ij}) , (v_{ij}, p_{ij}) , (p_{ij}, j) , respectively. The constraints (23) represent non-negativity and

flow efficiency constraints. The constraints in (24), (25) represent flow conservation constraints, where the exogenous arrival flow rates for nodes v_{ij}, p_{ij} are zero. The constraints in (26) represent the physical link constraint for virtual link (v_{ij}, p_{ij}) , which equals the link constraint for the real link (i, j) in the original system. Note that there is no explicit link constraints for (i, v_{ij}) and (p_{ij}, j) , since the transfer of packets over these links happen at the same node, and there is no physical link constraint on them.

Lemma 3 *The network capacity region for the virtual network Λ' defined in Theorem 3 is equal to that for the original system Λ defined in Theorem 1.*

Proof: First, we show that if $\lambda \in \Lambda'$, then it must lie in Λ as well. This can be shown directly by letting $\mu_{ij}^c = \mu_{ij}^{2c}$. Thus, we have $\mu_{ij}^{1c} \leq \mu_{ij}^c \leq \mu_{ij}^{3c}$. Plugging into (24), we have (3), i.e., if λ satisfies the constraints in (23)-(28), it must satisfy (2)-(7) as well. Thus $\lambda \in \Lambda$.

The other direction can be shown in the following way: we prove that for any $\lambda + \epsilon \in \Lambda$, $\lambda + \frac{\epsilon}{2d+1} \in \Lambda'$, where d is the maximum degree of the network.

Since $\lambda + \epsilon \in \Lambda$, we have

$$\sum_l \mu_{li}^c + \lambda_i^c + \epsilon \leq \sum_j \mu_{ij}^c, \quad \forall i \neq c. \quad (29)$$

By letting $\mu_{ij}^{2c} = \mu_{ij}^c$, we have that (26)-(28) satisfied. At the same time, we let $\mu_{ij}^{1c} + \epsilon_1 = \mu_{ij}^{3c} - \epsilon_1 = \mu_{ij}^{2c}$, and plug them into (29), which gives

$$\sum_l (\mu_{li}^{3c} - \epsilon_1) + \lambda_i^c + \epsilon \leq \sum_j (\mu_{ij}^{1c} + \epsilon_1), \quad \forall i \neq c. \quad (30)$$

Therefore,

$$\sum_l \mu_{li}^{3c} + \lambda_i^c + \epsilon - 2\epsilon_1 \leq \sum_j \mu_{ij}^{1c}, \quad \forall i \neq c. \quad (31)$$

By letting $\epsilon - 2\epsilon_1 = \epsilon_1$, we have

$$\sum_l \mu_{li}^{3c} + \lambda_i^c + \epsilon_1 \leq \sum_j \mu_{ij}^{1c}, \quad \forall i \neq c \quad (32)$$

$$\mu_{ij}^{1c} + \epsilon_1 = \mu_{ij}^{2c}, \quad \mu_{ij}^{2c} + \epsilon_1 = \mu_{ij}^{3c}. \quad (33)$$

Thus, we have $\lambda + \epsilon_1 \in \Lambda'$. As $\epsilon \rightarrow 0$, ϵ_1 approaches zero as well. Thus, $\Lambda = \Lambda'$. ■

Theorem 4 *For $\eta \geq 1$, the proposed algorithm stabilizes any rate vector satisfying $\lambda + \epsilon \in \Lambda$. The average expected queue backlog in the system is upper bounded by*

$$\frac{(2d+1)(KN(d+A_{max})^2 + 2N + (2\eta+1)KNd)}{\epsilon}.$$

The proof of the theorem is given in Appendix D. Since the upper bound is monotonically increasing in η , we can always set η to 1 to achieve a better delay performance.

The algorithm updates every slot. This avoids the delay caused by infrequent policy updating in the T -slot algorithm. On the other hand, we introduce virtual queues in the system. Since the differential backlog in Step 1) of the algorithm is not the physical differential backlog between nodes (i, j) in the

real system, the inaccuracy of the queue length information can, potentially, increase the average backlog in the system. This is reflected by the $2d+1$ factor in the upper bound. But for arrival rate vectors that are close to the boundary of network capacity region, the T -slot algorithm can only stabilize the system if T is large. In such situation thus the virtual-queue based algorithm attains a better delay performance.

The algorithm exhausts the maximum stability region of the network without any pre-specified parameter depending on traffic statistics. Compared to the T -slot algorithm where the stabilizing parameter T actually depends on how close a rate vector λ gets to the boundary of the network stability region Λ , this is a big advantage.

VI. DISCUSSIONS

A. Enhanced Virtual Queue Based Algorithm

According to Theorem 2, the average expected backlog in the system under T -slot algorithm is $O(T^2)$, which indicates poor delay performance when T is large. The virtual queue based algorithm avoids the long delay caused by the infrequent updating of queue length information. However, because of the introduction of virtual nodes, packets accumulate in virtual queues over weak links, which negatively impacts delay performance, especially when the system is lightly loaded. In order to improve delay performance, we proposed to enhance the virtual queue based algorithm by adjusting the term associated with V_{ij} in the Lyapunov function.

Define a modified Lyapunov function

$$L(\mathbf{U}(t), \mathbf{V}(t), \mathbf{P}(t)) = \sum_{c,i} (U_i^c(t))^2 + \sum_{c,(i,j)} (V_{ij}^c(t) + \gamma/r_{ij})^2 + \sum_{c,(i,j)} (P_{ij}^c(t) - \eta)^2. \quad (34)$$

Compared with (19), we have added a term γ/r_{ij} to each $V_{ij}^c(t)$ in (34). This is equivalent to add queue backlog γ/r_{ij} in the virtual queue. Following an analysis similar to that of the virtual queue algorithm (cf. Appendix C), we can show that under constraints (14) and (15), the sample path Lyapunov drift satisfies

$$\begin{aligned} \Delta(t) \leq & 2 \sum_{c,i} U_i^c(t) A_i^c(t) - 2 \sum_{c,i,j} [V_{ij}^c(t) + \gamma/r_{ij} - P_{ij}^c(t)] \beta_{ij}^c(t) \\ & - 2 \sum_{c,i,j} [U_i^c(t) - (V_{ij}^c(t) + \gamma/r_{ij})] \beta_{ij}^{1c}(t) \\ & - 2 \sum_{c,i,j} [(P_{ij}^c(t) - \eta) - U_j^c(t)] f_{ij}^c(t) + \alpha_3 \end{aligned}$$

where α_3 is a positive constant. In order to minimize the one-step Lyapunov drift, we substitute the following “modified” step M1) for step 1) of the virtual queue based algorithm as follows:

- M1) **Find per-link backpressure.** At the beginning of a timeslot, node i checks its neighbors and computes the differential backlogs. We compute the weight for the link between node i and the first virtual node, and, separately, the weight for the link between the two virtual nodes.

Specially, the weight for control action β_{ij}^{1c} is computed as

$$W_{ij}^{1c}(t) = [U_i^c(t) - (V_{ij}^c(t) + \gamma/r_{ij}) + (V_{ij}^c(t) + \gamma/r_{ij} - P_{ij}^c(t))r_{ij}]^+,$$

and the weight for control action β_{ij}^{2c} is computed as

$$W_{ij}^{2c}(t) = [V_{ij}^c(t) + \gamma/r_{ij} - P_{ij}^c(t)]^+ r_{ij}.$$

The weight of commodity c over link (i, j) is $W_{ij}^c(t) = \max\{W_{ij}^{1c}(t), W_{ij}^{2c}(t)\}$. The weight for link (i, j) is

$$W_{ij}(t) = \max_c W_{ij}^c(t),$$

and the optimal commodity

$$c_{ij}^* = \arg \max_c W_{ij}^c(t).$$

The rest of the steps remain the same as in Sec. V-B. Following similar steps as in the proof of Theorem 4, we can show that the enhanced version also achieves the maximum stability region. The intuition is, in heavily backlogged regime $V_{ij}^c(t) \gg \gamma/r_{ij}$, therefore the added queue backlog is negligible and doesn't impact the stability of the queue.

As in the virtual queue based algorithm, in the enhanced algorithm, we also set $\eta = 1$ to guarantee that $f_{ij}^c(t) = 1$ only when $P_{ij}^c(t) \geq 1$. We now discuss the effect of the γ parameter. We set $\gamma = 1/2$. By setting γ to be some positive value, the system adds some *virtual* backlog to buffer V_{ij} , thus preventing packets from entering the empty buffers over the weak links when the system starts from an initial empty state. It also increases backpressure between V_{ij} and P_{ij} . Therefore, packets tend to be pushed through links more quickly, and the decoding time is shortened accordingly. Besides, in the modified Lyapunov function, we select weights for the *virtual* backlogs of virtual queues as the inverse of link rates. The reason for such selection is that the number of slots required to deliver a packet through a link is equal to the inverse of the link rate. We aim to capture different delay effects over different links through this adjustment. The intuition behind the enhanced algorithm is that, when the system is lightly loaded, passing packets only through strong links can support the traffic load while still providing good delay performance. Therefore, using weak links is not necessary, and using strong links is preferable. Setting the virtual backlog length to be γ/r_{ij} forces packets to select strong links and improve the delay performance of the virtual queue based algorithm in the low traffic regime. When the system is heavily loaded and strong links cannot support all traffic flows, the differential backlogs over certain strong links eventually decrease, and weak links start to be used. The enhanced algorithm essentially is a hybrid of the classic backpressure algorithm ($T = 1$) and virtual queue based algorithm. It always stabilizes the system, and automatically adjusts the portion of slots that the system operates under each algorithm to maintain a good delay performance.

B. Dependent Link Rates

For the simplicity of analysis, in previous sections we assume that the link capacity for a given transmitter-receiver pair is fixed. We can easily relax this assumption and generalize the system model by assuming that the link rates are not fixed but are rather a function of the chosen activation pattern. Specifically, to be better able to capture the effects of interference, for link pair (i, j) we assume the link rate under activation pattern s is $r_{ij}(s)$. Then, the network capacity region under the new assumption is characterized by the same inequalities in Theorem 1 except that in eqn. (4) $r_{ij}(s)$ is used instead of r_{ij} .

The scheduling algorithms should be adjusted accordingly in order to achieve the network capacity region. E.g., for the T -slot algorithm, in each updating slot, after determining the maximizing commodity c_{ij}^* for each link (i, j) , the system selects the activation pattern, as well as the corresponding forwarder for each active transmitter, to maximize

$$\max_s \sum_{i \in s, j \in \mathcal{N}(i)} [Q_i^{c^*}(t) - Q_j^{c^*}(t)]^+ r_{ij}(s).$$

The remaining steps remain the same. For the virtual queue based algorithm, the maximizing commodity should be jointly selected with the activation pattern and the corresponding forwarders at the same time.

C. Distributed Implementation

The T -slot algorithm and the virtual queue based algorithm presented in the previous sections involving solving a constrained optimization problem (max-weight matching) in a centralized fashion. Here we consider distributed implementations. We assume nodes have information of the link rates between themselves and their neighbors, and the queue backlogs of their neighbors.

In interference-free networks where nodes can transmit simultaneously without interfering with each other, minimizing the Lyapunov drift function can be decomposed into local optimization problems. Each node individually makes the transmission decision based only on locally available information.

In networks with non-trivial interference constraints, we can use the message passing algorithm proposed in [19] to select the activation pattern in a distributed way. First, each node selects its forwarder based on its local information. Then, the system uses the belief propagation based message-passing algorithm to select the optimal activation pattern that minimizes the Lyapunov drift. If the underlying interference graph is bipartite, [19] shows that the message-passing algorithm always converges to the optimal solution. Another possible approach is to develop carrier sensing based randomized scheduling algorithms. Carrier sensing based distributed algorithms are discussed in [20], [21], etc. The throughput optimality of these algorithms are established under certain assumptions. Some other distributed implementations of backpressure algorithms are discussed in Chapter 4.8 of [17].

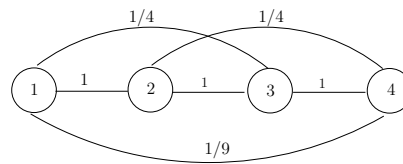


Fig. 2. The 4-node network used to compare the T -slot algorithm and the virtual queue based algorithm. The number labeling each link is the rate of that link.

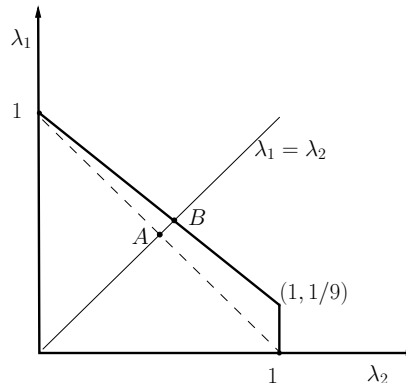


Fig. 3. The maximum stability region of the 4-node network. The inside triangular region is the network capacity region when MIA is not exploited.

VII. SIMULATION RESULTS

In this section we present detailed simulation results. These results illustrate the basic properties of the algorithms.

A. A single-commodity scenario

First consider the 4-node wireless network shown in Fig. 2 where the links with nonzero rates are shown. We consider a single commodity scenario where new packets destined for node 4 arrive at node 1 and node 2 according to independent Bernoulli processes with rate λ_1 and λ_2 , respectively. Node 3 does not have incoming packets. It acts purely as a relay. We assume that the system does not have any activation constraints, i.e., all nodes can transmit simultaneously without causing each other interference.

The maximum stability region Λ , shown in Fig. 3, is the union of rate pairs (λ_1, λ_2) defined according to Theorem 1. If mutual information accumulation is not allowed, the corresponding maximum stability region is indicated by the dashed triangle inside Λ . This follows because when weak links are not utilized, the only route from node 1 to node 4 is through node 2, thus the sum of arrival rates from node 1 and 2 cannot exceed link rate 1. When mutual information accumulation is exploited, the weak link from node 1 to node 4 with rate $1/9$ can be utilized, expanding the stability region.

We first compare the performance of the T -slot algorithm for different values of T . For each T , we conduct the simulation for arrival rates $\lambda_1 = \lambda_2 = \lambda$ ranging from 0 to 0.55. The resulting average backlog curve is shown in Fig. 4. When $T = 1$, the weak links cannot be utilized, and the algorithm can only stabilize the arrival rates up to $\lambda = 1/2$, which is point A in Fig. 3. When $T = 9$, the reciprocal of the link rate $1/9$, the algorithm can stabilize arrival rates up to $\lambda = 9/17$, corresponding to point B in Fig. 3. In this case,

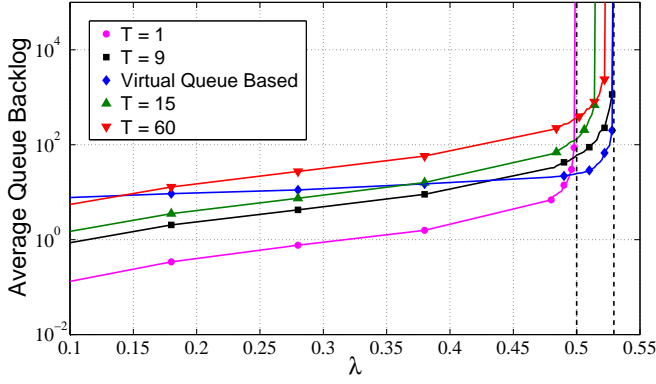


Fig. 4. Comparison of average backlog in the system under the algorithms.

all of the partial packets transferred over weak link (1, 4) are eventually decoded, and that weak link is fully utilized. This is a special scenario since the value of T perfectly matches the rate of weak link. For larger networks that consist of many weak links, selecting T to match all of the weak links may be prohibitive, since such value can be very large. Except for such perfect matching scenarios, for more general values of T , the weak link is partially utilized, therefore, the maximum λ the algorithm can stabilize is some value between $1/2$ and $9/17$. In general, a larger T stabilizes larger arrival rates, and results in an increased average backlog in the system. This is illustrated by curves with $T = 15, 60$ in Fig. 4.

Fig. 4 also plots the performance of the virtual queue based algorithm. The system can be stabilized up to $\lambda = 9/17$ under the virtual queue based algorithm. Compared with the T -slot algorithm, the virtual queue based algorithm attains a much better delay performance for large value of λ , i.e., when the system is in heavy traffic regime. It dominates even the curve with $T = 9$ at high rates. For small values of λ , the virtual queue based algorithm has a worse performance in terms of delay. This is because the algorithm requires the virtual queues to build up certain lengths in order to push packets through the weak links. The virtual queue based algorithm has relatively constant delay performance for $\lambda \in [0, 1/2]$, while under the T -slot algorithm, the average backlog increases monotonically with λ .

B. A multi-commodity scenario

Next we consider the 10-node network shown in Fig. 5. We consider a multi-commodity scenario in which packets destined for node 10 arrive at node 1 and packets destined for node 9 arrive at node 2. Arrivals are distributed according to two independent Bernoulli processes with rate λ_1 and λ_2 , respectively. We assume that the system does not have any activation constraints so that all nodes can transmit simultaneously without causing interference.

The maximum stability region Λ is shown in Fig. 6. If mutual information accumulation is not allowed, the corresponding maximum stability region is the dashed triangle inside Λ . This follows because when weak links are not used, the routes from node 1 to node 10 and the routes from node 2 to node 9 must pass through link (4, 7), thus the sum of arrival

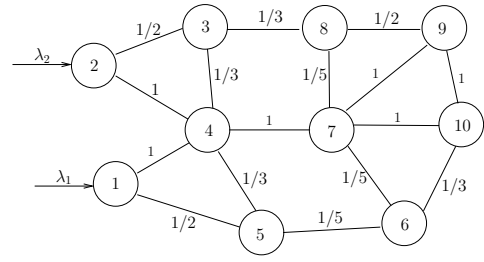


Fig. 5. The 10-node network used to compare the T -slot algorithm and the virtual queue based algorithm. The number labeling each link is the rate of that link.

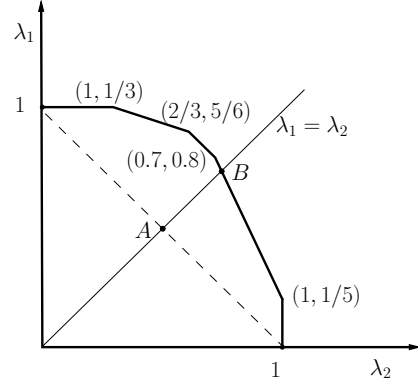


Fig. 6. The maximum stability region of the 10-node network, where $B = (11/15, 11/15)$. The dashed triangular region is the network capacity region when MIA is not exploited.

rates from node 1 and 2 cannot exceed that link rate. When mutual information accumulation is exploited, weak links can be utilized and form additional routes $1 \rightarrow 5 \rightarrow 6 \rightarrow 10$, $2 \rightarrow 3 \rightarrow 8 \rightarrow 9$, thus an expanded stability region can be achieved.

We first compare the performance of the T -slot algorithm for different values of T . For each T , we conduct the simulation for arrival rates $\lambda_1 = \lambda_2 = \lambda$ ranging from 0 to 0.75. The resulting average backlog curve is shown in Fig. 7. When $T = 1$, the weak links are not utilized, so the algorithm can only stabilize the arrival rates up to $\lambda = 1/2$, which is point A in Fig. 6. When $T = 30$, which is the reciprocal of the link rate product $\frac{1}{2} \frac{1}{3} \frac{1}{5}$ and perfectly matches the rates of weak links, the algorithm can stabilize the arrival rates up to $\lambda = 11/15$, corresponding to point B in Fig. 6. For more general values of T , the weak links are partially utilized, therefore, the maximum λ the algorithm can stabilize is some value between $1/2$ and $11/15$. In general, a larger T stabilizes larger arrival rates, and results in an increased average backlog in the system. This is illustrated by curves with $T = 5, 17$ in Fig. 7. In order to achieve the boundary point, a large T is required in general.

In Fig. 6, we also present performance results from simulation of the virtual queue based algorithm. As expected, the system can be stabilized up to the edge of the stability region, $\lambda = 11/15$. Compared with the T -slot algorithms, the virtual queue based algorithm attains a much better delay

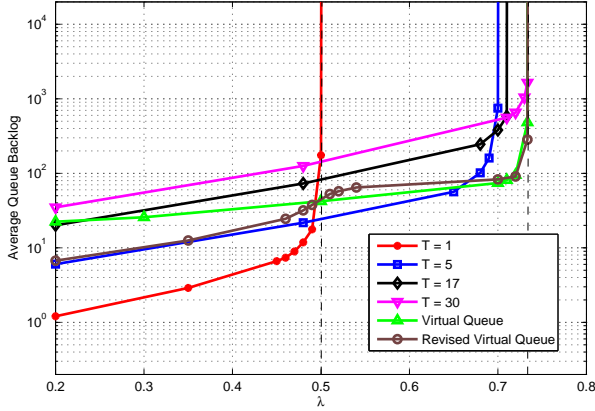


Fig. 7. Comparison of average backlog in the system under the algorithms.

performance in heavy traffic regime. It dominates the curve with $T = 30$ over the displayed rate region. Similar to the single-commodity scenario, for small values of λ , the virtual queue based algorithm does not show much advantage in terms of delay. Finally, we also provide simulation results for the enhanced virtual queue based algorithm in Fig. 7. The enhanced algorithm stabilize the full spectrum of arrival rate, i.e., every input rate vector up to $\lambda = 11/15$. The delay performance in the light traffic regime ($\lambda < 1/2$) is improved under the enhanced version, with a smaller penalty of delay in the heavy traffic regime. The delay performance transition around $\lambda < 1/2$ can be explained by the hybrid of and automatic adjustment between the classic backpressure ($T = 1$) and virtual queue based algorithm. In the simulation, we set $\gamma = 1/2$. The value of γ decides the tradeoff between the delay performance in the light traffic regime and the delay performance in the heavy traffic regime.

VIII. CONCLUSIONS

In this paper, we analyzed the optimal routing and scheduling policy when mutual information accumulation is exploited at the physical layer in a wireless network. We first characterized the maximum stability region under the natural assumption of policy space, which is shown to surpass the network capacity region when mutual information accumulation is not allowed. Two scheduling policies are proposed to cope with the decoding process introduced by mutual information accumulation. The T -slot algorithm and the virtual queue based algorithm can both achieve the maximum stability region but the latter has significantly reduced delay in heavy traffic regime. We also compared the performance under these two policies analytically and numerically.

APPENDIX A

PROOF OF THE NECESSITY PART OF THEOREM 1

Suppose that a stabilizing control strategy exists. It is possible that the strategy may use redundant packet transfers, i.e., allowing multiple copies of a packet to exist at the same time in the network. However, under assumption A3, each packet can only have a single parent, i.e., once a node starts to accumulate information for a packet, it can only decode

the packet if the total received information from a single transmitter exceeds the amount of information contained in that packet.

Define $X_i^c(t)$ as the total number of packets of commodity c that have arrived at node i up to slot t . Define $\mathcal{D}^c(t)$ as the set of distinct packets of commodity c delivered to the destination node c over $[0, t)$, and $D^c(t) = |\mathcal{D}^c(t)|$ be the total number of such distinct packets. Then, we have $D^c(t) = \sum_{i=1}^n Y_i^c(t)$, where $Y_i^c(t)$ is defined in Section III. If multiple copies of a packet depart from the system, we only count the first delivered copy. In a stable system, for any traffic flow of commodity c entering source node i , the average number of distinct packets delivered to destination node c in each slot must be equal to its arrival rate, thus, we have

$$\lim_{t \rightarrow \infty} \frac{Y_i^c(t)}{t} = \lim_{t \rightarrow \infty} \frac{X_i^c(t)}{t} = \lambda_i^c. \quad (35)$$

For each distinct delivered packet k , there is a single routing path this packet took from its origin node to its destination node. Denote $D_{ij}^c(t)$ as the total number of distinct packets in $\mathcal{D}^c(t)$ transmitted from node i to node j over $[0, t)$. We have

$$\sum_{l=1, l \neq i}^n D_{li}^c(t) + Y_i^c(t) = \sum_{j=1, j \neq i}^n D_{ij}^c(t). \quad (36)$$

Define $T_s(t)$ to be the number of slots within which the system operates with activation pattern $s \in \mathcal{S}$ up to time t , and let $T_{ij}^c(s, t)$ be the number of slots that link (i, j) is active and transmitting a packet in $\mathcal{D}^c(t)$ under activation pattern s up to time t . Therefore, we have

$$D_{ij}^c(t) = \sum_{s \in \mathcal{S}} T_{ij}^c(s, t) r_{ij}, \quad (37)$$

Thus,

$$\frac{D_{ij}^c(t)}{t} = \sum_{s \in \mathcal{S}} \frac{T_s(t)}{t} \frac{T_{ij}^c(s, t)}{T_s(t)} r_{ij}. \quad (38)$$

Define $\mu_{ij}^c(t) = \frac{D_{ij}^c(t)}{t}$, $\pi_s(t) = \frac{T_s(t)}{t}$ and $\theta_{ij}^c(s, t) = \frac{T_{ij}^c(s, t)}{T_s(t)}$. We note that since we can deliver at most one packet to i from j in each time slot,

$$0 \leq \mu_{ij}^c(t) \leq 1, \quad \mu_{ii}^c(t) = 0, \quad \mu_{c,i}^c(t) = 0. \quad (39)$$

Because only one activation pattern is allowed per slot, we have

$$\sum_{s \in \mathcal{S}} \pi_s(t) = 1. \quad (40)$$

On the other hand, since a node can only transmit a single packet in any slot, and we only count distinct copies of packets, then, if node i is transmitting a packet of commodity c at time t , at most one of the received copies at its neighbors can be counted as a distinct packet in $\mathcal{D}^c(t)$. Thus, we have

$$\sum_{c,j} \theta_{ij}^c(s, t) \leq 1 \quad \text{if } i \in \mathcal{S}. \quad (41)$$

We can always make inequality (41) tight by restricting to the policy space that a node can only transmit when it is necessary, i.e., if node i 's transmission at time t does not contribute to

the delivery of any distinct packet to its destination, it should keep silent and be removed from the activation pattern at time t . The remaining active nodes form another valid activation pattern, and gives $T_s(t) = \sum_{c,j} T_{ij}^c(s, t)$ for every $i \in s$.

These constraints define a closed and bounded region with finite dimension, thus there must exist an infinite subsequence \tilde{t} over which the individual terms converges to points μ_{ij}^c , π_s and $\theta_{ij}^c(s)$ that also satisfy the inequalities (39)-(41):

$$\lim_{\tilde{t} \rightarrow \infty} \mu_{ij}^c(\tilde{t}) = \mu_{ij}^c, \quad (42)$$

$$\lim_{\tilde{t} \rightarrow \infty} \pi_s(\tilde{t}) = \pi_s, \quad (43)$$

$$\lim_{\tilde{t} \rightarrow \infty} \theta_{ij}^c(s, \tilde{t}) = \theta_{ij}^c(s). \quad (44)$$

Furthermore, using (35) in (36) and taking $\tilde{t} \rightarrow \infty$ yields

$$\sum_l \mu_{li}^c + \lambda_i^c = \sum_j \mu_{ij}^c, \quad \forall i \neq c. \quad (45)$$

This proves the result.

APPENDIX B PROOF OF THEOREM 2

Our algorithm always transmit packets from the head of $Q_i^c(t)$, and there is at most one full copy of each packet in the network. Therefore, without worry of confusion, in the following analysis we drop packet index k and use commodity index c instead as the superscript of control actions $\{\beta_{ij}^{(k)}\}$ and $\{f_{ij}^{(k)}\}$.

First, define the Lyapunov function

$$L(\mathbf{Q}(t)) = \sum_{c,i} (Q_i^c(t))^2,$$

and the T -slot sample path Lyapunov drift as

$$\Delta_T(t) := L(\mathbf{Q}(t+T)) - L(\mathbf{Q}(t)).$$

Then, we have the following Lemma.

Lemma 4 Assume the system changes its policy every T slots starting at $t = 0$. For all $t = 0, T, 2T, \dots$ and all possible values of $\mathbf{Q}(t)$, under a given policy $\{\beta_{ij}^c(t)\}$, we have

$$\begin{aligned} \Delta_T(t) &\leq \sum_{c,i} -2Q_i^c(t) \left(T \sum_j \beta_{ij}^c(t) r_{ij} - T \sum_l \beta_{li}^c(t) r_{li} \right. \\ &\quad \left. - \sum_{\tau=0}^{T-1} A_i^c(t+\tau) - 1 \right) + \alpha_1, \end{aligned} \quad (46)$$

where

$$\alpha_1 = KNT^2(\mu_{max} + A_{max})^2 + NT^2. \quad (47)$$

Proof: Under any given policy, the queue length evolves according to

$$\begin{aligned} Q_i^c(t+1) &= \left(Q_i^c(t) - \sum_j \beta_{ij}^c(t) f_{ij}^c(t) \right)^+ \\ &\quad + \sum_l \beta_{li}^c(t) f_{li}^c(t) + A_i^c(t). \end{aligned} \quad (48)$$

Considering the policy which updates at $t = 0, T, 2T, \dots$, we have

$$\begin{aligned} Q_i^c(t+T) &\leq \left(Q_i^c(t) - \sum_{\tau=0}^{T-1} \sum_j \beta_{ij}^c(t) f_{ij}^c(t+\tau) \right)^+ \\ &\quad + \sum_{\tau=0}^{T-1} \sum_l \beta_{li}^c(t) f_{li}^c(t+\tau) + \sum_{\tau=0}^{T-1} A_i^c(t+\tau) \end{aligned} \quad (49)$$

$$\begin{aligned} &\leq \left(Q_i^c(t) - \sum_j \beta_{ij}^c(t) \lfloor r_{ij} T \rfloor \right)^+ \\ &\quad + \sum_l \beta_{li}^c(t) \lfloor r_{li} T \rfloor + \sum_{\tau=0}^{T-1} A_i^c(t+\tau). \end{aligned} \quad (50)$$

In (49), we upper bound $Q_i^c(t+T)$ by moving the negative terms into the function $(\cdot)^+$. This follows from the facts that

$$\max[a+b-c, 0] \leq \max[a-c, 0] + b$$

$$\max[\max[a, 0] - c, 0] = \max[a - c, 0] \text{ for } a, b, c \geq 0.$$

This is equivalent to letting node i transmit packets existing in $Q_i^c(t)$ only, i.e., even if there are some packets of commodity c arrive at node i in the epoch and all of the packets existing in $Q_i^c(t)$ have been cleared, they are not transmitted until next epoch. Since under the policy, these packets may be transmitted to next hop, the actual queue length $Q_i^c(t+T)$ can be upper bounded. Eqn. (50) follows from the fact that over the T -slot window, the successfully delivered packets (including dummy packets) from node i to node j is $\lfloor T \beta_{ij}^c(t) r_{ij} \rfloor$, and recall $\beta_{ij}^c(t)$ is held constant for the whole epoch. Since both sides of the inequality are positive, it holds for the square of both sides, thus,

$$\begin{aligned} (Q_i^c(t+T))^2 &\leq \left(Q_i^c(t) - \sum_j \beta_{ij}^c(t) \lfloor r_{ij} T \rfloor \right)^2 \\ &\quad + 2Q_i^c(t) \left(\sum_l \beta_{li}^c(t) \lfloor r_{li} T \rfloor + \sum_{\tau=0}^{T-1} A_i^c(t+\tau) \right) \\ &\quad + \left(\sum_l \beta_{li}^c(t) \lfloor r_{li} T \rfloor + \sum_{\tau=0}^{T-1} A_i^c(t+\tau) \right)^2 \end{aligned} \quad (51)$$

$$\begin{aligned} &\leq (Q_i^c(t))^2 - 2Q_i^c(t) \left(\sum_j \beta_{ij}^c(t) \lfloor r_{ij} T \rfloor \right. \\ &\quad \left. - \sum_l \beta_{li}^c(t) \lfloor r_{li} T \rfloor - \sum_{\tau=0}^{T-1} A_i^c(t+\tau) \right) + C_i^c, \end{aligned} \quad (52)$$

where

$$\begin{aligned} C_i^c &= \left(\sum_l \beta_{li}^c(t) \lfloor r_{li} T \rfloor + \sum_{\tau=0}^{T-1} A_i^c(t+\tau) \right)^2 \\ &\quad + \left(\sum_j \beta_{ij}^c(t) \lfloor r_{ij} T \rfloor \right)^2. \end{aligned}$$

We use $Q_i^c(t)$ instead of $\left(Q_i^c(t) - \sum_j \beta_{ij}^c(t) \lfloor r_{ij} T \rfloor \right)^+$ for the cross term in (51). Since the former is always greater than the

latter, the inequality (51) holds. Therefore, we have

$$\Delta_T(t) \leq \sum_{c,i} -2Q_i^c(t) \left(\sum_j \beta_{ij}^c(t) \lfloor r_{ij} T \rfloor - \sum_l \beta_{li}^c(t) \lfloor r_{li} T \rfloor - \sum_{\tau=0}^{T-1} A_i^c(t+\tau) \right) + \sum_{c,i} C_i^c \quad (53)$$

$$\leq \sum_{c,i} -2Q_i^c(t) \left(\sum_j \beta_{ij}^c(t) (r_{ij} T - 1) - \sum_l \beta_{li}^c(t) r_{li} T - \sum_{\tau=0}^{T-1} A_i^c(t+\tau) \right) + \sum_{c,i} C_i^c \quad (54)$$

$$\leq \sum_{c,i} -2Q_i^c(t) \left(T \sum_j \beta_{ij}^c(t) r_{ij} - T \sum_l \beta_{li}^c(t) r_{li} - \sum_{\tau=0}^{T-1} A_i^c(t+\tau) - 1 \right) + \sum_{c,i} C_i^c \quad (55)$$

where $\Delta_T(t)$ is the T -slot Lyapunov drift, (54) follows from the fact that $x - 1 < \lfloor x \rfloor \leq x$, and (55) follows from the fact that $\sum_j \beta_{ij}^c(t) \leq 1$. Based on the assumptions that $A_i^c(t) \leq A_{max}$, the maximum number of decoded packets at a node is upper bounded by μ_{max} , and the constraint (8), we have

$$\sum_{c,i} C_i^c \leq KNT^2(\mu_{max} + A_{max})^2 + NT^2 := \alpha_1$$

The proof is completed. ■

Lemma 5 For a given $\mathbf{Q}(t)$ on slot $t = 0, T, 2T, \dots$, under the T -slot algorithm, the T -slot Lyapunov drift satisfies

$$\Delta_T(t) \leq \sum_{c,i} -2Q_i^c(t) \left(T \sum_j \hat{\beta}_{ij}^c(t) r_{ij} - T \sum_l \hat{\beta}_{li}^c(t) r_{li} - \sum_{\tau=0}^{T-1} A_i^c(t+\tau) - 1 \right) + \alpha_1, \quad (56)$$

where $\{\hat{\beta}_{ij}^c(t)\}$ are any alternative (possibly randomized) decisions that satisfy (8)-(9). Further more, we have

$$\mathbb{E}\{\Delta_T(t)|\mathbf{Q}(t)\} \leq \mathbb{E}\left\{ \sum_{c,i} -2Q_i^c(t) \left(T \sum_j \hat{\beta}_{ij}^c(t) r_{ij} - T \sum_l \hat{\beta}_{li}^c(t) r_{li} - T\lambda_i^c - 1 \right) \middle| \mathbf{Q}(t) \right\} + \alpha_1 \quad (57)$$

Proof: Given $\mathbf{Q}(t)$, the T -slot algorithm makes decisions to minimize the right hand side of (46). Therefore, inequality (56) holds for all realizations of the random quantities, and hence also holds when taking conditional expectations of both sides. Thus, we have (57). ■

Corollary 1 A rate matrix $\{\lambda + \epsilon\}$ is in the capacity region Λ if and only if there exists a stationary (possibly randomized) algorithm that chooses control decisions subject to constraint (8)-(9) and independent of current queue backlog to yield

$$\mathbb{E}\left\{ \sum_j \beta_{ij}^c r_{ij} - \sum_l \beta_{li}^c r_{li} - \lambda_i^c \right\} \geq \epsilon \quad \forall i \neq c. \quad (58)$$

Proof: The result is an immediate consequence of Theorem 1. The intuition is to think $\mathbb{E}\{\beta_{ij}^c\} r_{ij}$ as μ_{ij}^c in (2)-(7). The necessary part is obtained directly. The sufficient part will be shown in the following section. ■

For any $\{\lambda + \epsilon\} \in \Lambda$, Lemma 5 shows that the T -slot algorithm minimizes the right hand side of (57) for any alternative policy satisfying (8)-(9). On the other hand, Corollary 1 implies that such policy can be constructed in a randomized way which is independent of current queue status in the network and satisfying (58). Combining Lemma 5 and Corollary 1, we have

$$\begin{aligned} \mathbb{E}\{\Delta_T(t)|\mathbf{Q}(t)\} &\leq \sum_{c,i} -2Q_i^c(t) \left(\left(\hat{\beta}_{ij}^c(t) r_{ij} - \sum_l \hat{\beta}_{li}^c(t) r_{li} - \lambda_i^c \right) T - 1 \right) + \alpha_1 \\ &\leq \sum_{c,i} -2Q_i^c(t) (\epsilon T - 1) + \alpha_1 \end{aligned}$$

Taking expectations of the above inequality over the distribution of $\mathbf{Q}(t)$, we have

$$\mathbb{E}\{\Delta_T(t)\} \leq \sum_{c,i} -2\mathbb{E}\{Q_i^c(t)\} (\epsilon T - 1) + \alpha_1. \quad (59)$$

Summing terms over $t = 0, T, \dots, (M-1)T$ for positive integer M yields

$$\begin{aligned} &\frac{\mathbb{E}\{L(\mathbf{Q}(MT)) - L(\mathbf{Q}(0))\}}{M} \\ &\leq \frac{\sum_{m=0}^{M-1} \sum_{c,i} -2\mathbb{E}\{Q_i^c(mT)\} (\epsilon T - 1)}{M} + \alpha_1, \end{aligned}$$

i.e.,

$$\begin{aligned} \sum_{m=0}^{M-1} \sum_{c,i} \mathbb{E}\{Q_i^c(mT)\} &\leq \frac{L(\mathbf{Q}(0)) - L(\mathbf{Q}(MT)) + M\alpha_1}{2(\epsilon T - 1)} \\ &\leq \frac{L(\mathbf{Q}(0)) + M\alpha_1}{2(\epsilon T - 1)}. \end{aligned} \quad (60)$$

We drop $L(\mathbf{Q}(MT))$ in (60) since $L(\mathbf{Q}(MT)) \geq 0$ based on the definition of Lyapunov function. On the other hand, for $t = 0, T, 2T, \dots$ and $0 < \tau < T$, we have

$$Q_i^c(t+\tau) \leq Q_i^c(t) + (\mu_{max} + A_{max})\tau \quad (61)$$

where A_{max} is the maximum arrival rates and μ_{max} is the maximum number of decoded packets in a slot for any node. Therefore,

$$\begin{aligned} \sum_{m=0}^{M-1} \sum_{\tau=1}^{T-1} \sum_{c,i} \mathbb{E}\{Q_i^c(mT+\tau)\} &\leq \sum_{m=0}^{M-1} \sum_{c,i} (T-1) \mathbb{E}\{Q_i^c(mT)\} \\ &\quad + MT(T-1)KN(\mu_{max} + A_{max})/2 \end{aligned}$$

Combining with (60), we have

$$\begin{aligned} & \frac{1}{MT} \sum_{m=0}^{M-1} \sum_{\tau=0}^{T-1} \sum_{c,i} \mathbb{E}\{Q_i^c(mT + \tau)\} \\ & \leq \frac{1}{M} \sum_{m=0}^{M-1} \sum_{c,i} \mathbb{E}\{Q_i^c(mT)\} + KN(T-1)(\mu_{max} + A_{max})/2 \\ & \leq \frac{L(\mathbf{Q}(0)) + M\alpha_1}{2M(\epsilon T - 1)} + \frac{KN(T-1)(\mu_{max} + A_{max})}{2} \end{aligned}$$

Letting $M \rightarrow \infty$, we have

$$\begin{aligned} & \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{c,i} \mathbb{E}\{Q_i^c(\tau)\} \\ & \leq \frac{\alpha_1}{2(\epsilon T - 1)} + \frac{KN(T-1)(\mu_{max} + A_{max})}{2} \end{aligned}$$

Therefore, when $T > \frac{1}{\epsilon}$, the average expected backlog in the network is bounded. Thus, the system is stable. Since our algorithm operates under assumptions A1-A2, the total number of partial packets in the system, $\sum_{c,i} P_i^c(t)$, is always upper bounded by $\sum_{c,i} Q_i^c(t)$. Therefore, if the $\{Q_i^c(t)\}$ are stable, the $\{P_i^c(t)\}$ must be stable, and the overall system is stable.

APPENDIX C PROOF OF LEMMA 1

Based on (16)-(18), we have

$$\begin{aligned} (U_i^c(t+1))^2 & \leq (U_i^c(t))^2 + \left(\sum_j \beta_{ij}^{1c}(t)\right)^2 \\ & \quad - 2U_i^c(t) \left(\sum_j \beta_{ij}^{1c}(t) - \sum_l f_{li}^c(t) - A_i^c(t)\right) \\ & \quad + \left(\sum_l f_{li}^c(t) + A_i^c(t)\right)^2 \end{aligned} \quad (62)$$

$$\begin{aligned} (V_{ij}^c(t+1))^2 & \leq (V_{ij}^c(t))^2 + 2V_{ij}^c(t) (\beta_{ij}^{1c}(t) - \beta_{ij}^c(t)r_{ij}) \\ & \quad + (\beta_{ij}^{1c}(t) - \beta_{ij}^c(t)r_{ij})^2 \end{aligned} \quad (63)$$

$$\begin{aligned} (P_{ij}^c(t+1))^2 & \leq (P_{ij}^c(t))^2 + 2P_{ij}^c(t)(\beta_{ij}^c(t)r_{ij}(t) - f_{ij}^c(t)) \\ & \quad + (\beta_{ij}^c(t)r_{ij}(t) - f_{ij}^c(t))^2 \end{aligned} \quad (64)$$

$$P_{ij}^c(t+1) \geq P_{ij}^c(t) - f_{ij}^c(t) \quad (65)$$

Thus,

$$\begin{aligned} \Delta(t) & \leq \sum_{c,i} -2U_i^c(t) \left(\sum_j \beta_{ij}^{1c}(t) - \sum_l f_{li}^c(t) - A_i^c(t)\right) \\ & \quad + \sum_{c,i,j} 2V_{ij}^c(t) (\beta_{ij}^{1c}(t) - \beta_{ij}^c(t)r_{ij}) \\ & \quad + \sum_{c,i,j} [2P_{ij}^c(t)(\beta_{ij}^c(t)r_{ij}(t) - f_{ij}^c(t)) + 2\eta f_{ij}^c(t)] + C \end{aligned}$$

where

$$\begin{aligned} C & = \sum_{c,i} \left(\sum_j \beta_{ij}^{1c}(t)\right)^2 + \sum_{c,i} \left(\sum_l f_{li}^c(t) + A_i^c(t)\right)^2 \\ & \quad + \sum_{c,i,j} \left[(\beta_{ij}^{1c}(t) - \beta_{ij}^c(t)r_{ij})^2 + (\beta_{ij}^c(t)r_{ij}(t) - f_{ij}^c(t))^2\right] \end{aligned}$$

Because of constraints (14)-(15), we have

$$C \leq KN(d + A_{max})^2 + 2N + KNd := \alpha_2. \quad (66)$$

Combining items with respect to link (i, j) , we have (20).

APPENDIX D PROOF OF THEOREM 4

Corollary 2 A rate vector $\lambda + \epsilon$ is in the capacity region Λ' if and only if there exists a stationary (possibly randomized) algorithm that chooses control decisions (independent of current queue backlog) subject to constraints (14), to yield

$$\mathbb{E}\{\beta_{ij}^c\}r_{ij} = \mathbb{E}\{\beta_{ij}^{1c}\} + \epsilon \quad (67)$$

$$\mathbb{E}\{f_{ij}^c\} = \mathbb{E}\{\beta_{ij}^c\}r_{ij} + \epsilon \quad (68)$$

$$\mathbb{E}\left\{\sum_j f_{ij}^c - \sum_l \beta_{li}^{1c} - \lambda_i\right\} \geq \epsilon \quad \forall i \neq c \quad (69)$$

Proof: The result is an immediate consequence of Theorem 3. The intuition is to let $\mathbb{E}\{\beta_{ij}^c\}r_{ij} = \mu_{ij}^{2c}$, $\mathbb{E}\{\beta_{ij}^{1c}\} = \mu_{ij}^{1c}$ and $\mathbb{E}\{f_{ij}^c\} = \mu_{ij}^{3c}$. The necessary part is obtained directly. The sufficient part will be shown in the following proof. ■

Lemma 6 Under the virtual queue based algorithm,

$$\begin{aligned} & \sum_{c,i,j} \left\{ [U_i^c(t) - V_{ij}^c(t)]\beta_{ij}^{1c}(t) + [V_{ij}^c(t) - P_{ij}^c(t)]r_{ij}\beta_{ij}^c(t) \right. \\ & \quad \left. + [P_{ij}^c(t) - \eta - U_j^c(t)]f_{ij}^c(t) \right\} \\ & \geq \sum_{c,i,j} \left\{ [U_i^c(t) - V_{ij}^c(t)]\hat{\beta}_{ij}^{1c}(t) + [V_{ij}^c(t) - P_{ij}^c(t)]r_{ij}\hat{\beta}_{ij}^c(t) \right. \\ & \quad \left. + [P_{ij}^c(t) - \eta - U_j^c(t)]\hat{f}_{ij}^c(t) \right\} \end{aligned} \quad (70)$$

for any other binary control policy $\{\hat{\beta}_{ij}^{1c}, \hat{\beta}_{ij}^c, \hat{f}_{ij}^c\}$ satisfying (14).

Proof: This lemma is an immediate consequence of the fact that the virtual queue based algorithm maximizes the left hand side of (70) while satisfying (14). The constraint (15) is satisfied automatically. ■

Based on Lemma 1 (20) and Lemma 6, we have

$$\begin{aligned} & \mathbb{E}\{\Delta(t)|\mathbf{U}(t), \mathbf{V}(t), \mathbf{P}(t)\} \\ & \leq -2 \sum_{c,i,j} \mathbb{E}\left\{ [U_i^c(t) - V_{ij}^c(t)]\hat{\beta}_{ij}^{1c} + [V_{ij}^c(t) - P_{ij}^c(t)]r_{ij}\hat{\beta}_{ij}^c \right. \\ & \quad \left. + [P_{ij}^c(t) - \eta - U_j^c(t)]\hat{f}_{ij}^c(t) \right| \mathbf{U}(t), \mathbf{T}(t), \mathbf{P}(t) \Big\} \\ & \quad + 2 \sum_{c,i} U_i^c(t)\lambda_i^c + \alpha_2 \\ & \leq \mathbb{E}\left\{ \sum_{c,i} -2U_i^c(t) \left(\sum_j \hat{\beta}_{ij}^{1c}(t) - \sum_l \hat{f}_{li}^c(t) - \lambda_i^c(t)\right) \right. \\ & \quad \left. + \mathbb{E}\left\{ \sum_{c,i,j} 2V_{ij}^c(t) (\hat{\beta}_{ij}^{1c}(t) - \hat{\beta}_{ij}^c(t)r_{ij}) \right\} \right. \\ & \quad \left. + \mathbb{E}\left\{ \sum_{c,i,j} 2P_{ij}^c(t)(\hat{\beta}_{ij}^c(t)r_{ij}(t) - \hat{f}_{ij}^c(t)) + 2\eta\hat{f}_{ij}^c(t) \right\} \right\} + \alpha_2 \end{aligned}$$

Since for $\lambda + \epsilon \in \Lambda$, we have $\lambda + \epsilon/(2d+1) \in \Lambda'$, thus

$$\begin{aligned} & \mathbb{E}\{\Delta(t)|\mathbf{U}(t), \mathbf{V}(t), \mathbf{P}(t)\} \\ & \leq -2\left(\sum_{c,i} U_i^c(t) + \sum_{c,i,j} V_{ij}^c(t) + \sum_{c,i,j} P_{ij}^c(t)\right) \frac{\epsilon}{2d+1} \\ & \quad + 2\eta KNd + KN(d + A_{max})^2 + 2N + KNd. \end{aligned}$$

Therefore, the system is stable, and the average backlog is upper bounded by

$$\begin{aligned} & \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \left(\sum_{c,i} U_i^c(\tau) + \sum_{c,i,j} V_{ij}^c(\tau) + \sum_{c,i,j} P_{ij}^c(\tau) \right) \\ & \leq \frac{(2d+1)(KN(d + A_{max})^2 + 2N + (2\eta + 1)KNd)}{\epsilon} \end{aligned}$$

The proof is completed.

REFERENCES

- [1] S. Biswas and R. Morris, "ExOR: Opportunistic multi-hop routing for wireless networks," *ACM SIGCOMM*, vol. 35, pp. 133–144, Aug. 2005.
- [2] E. Rozner, J. Seshadri, Y. A. Mehta, and L. Qiu, "SOAR: Simple opportunistic adaptive routing protocol for wireless mesh networks," *IEEE Tran. Mobi. Comp.*, vol. 8, pp. 1622–1635, Dec. 2009.
- [3] M. J. Neely and R. Ugaonkar, "Optimal backpressure routing for wireless networks with multi-receiver diversity," *Proc. Conf. Info. Sci. Sys.*, pp. 18–25, Jan. 2006.
- [4] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automat. Contr.*, vol. 37, pp. 1936–1948, Apr. 1992.
- [5] —, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Trans. Inform. Theory*, vol. 39, pp. 466–478, Jun. 1993.
- [6] E. M. Yeh and R. A. Berry, "Throughput optimal control of cooperative relay networks," *IEEE Trans. Inform. Theory*, vol. 53, pp. 3827–3833, Oct. 2007.
- [7] S. C. Draper, L. Liu, A. F. Molisch, and J. S. Yedidia, "Cooperative routing for wireless networks using mutual-information accumulation," *IEEE Trans. Inform. Theory*, vol. 57, no. 8, pp. 5757–5762, 2011.
- [8] R. Ugaonkar and M. J. Neely, "Routing with mutual information accumulation in wireless networks," Tech. Rep., Aug. 2010.
- [9] Y. Liu, J. Yang, and S. C. Draper, "Exploiting route diversity in multi-packet transmission using mutual information accumulation," *Allerton Conf. on Commun., Control, and Computing*, Sept. 2011.
- [10] I. Maric and R. D. Yates, "Cooperative multihop broadcast for wireless networks," *IEEE J. Select. Areas Commun.*, vol. 22, pp. 1080–1088, Aug. 2004.
- [11] —, "Cooperative multicast for maximum network lifetime," *IEEE J. Select. Areas Commun.*, vol. 23, pp. 127–135, Jan. 2005.
- [12] J. Chen, L. Jia, X. Liu, G. Noubir, and R. Sundaram, "Minimum energy accumulative routing in wireless networks," *Proc. IEEE INFOCOM*, vol. 3, pp. 1875–1886, Mar. 2005.
- [13] M. Mitzenmacher, "Digital fountains: A survey and look forward," *Proc. IEEE Inform. Theory Workshop*, pp. 271–276, Oct. 2004.
- [14] M. Luby, "LT codes," *Proc. Symp. Foundation of Computer Science*, pp. 271–282, 2002.
- [15] E. Yeh and A. Cohen, "Throughput optimal power and rate control for queued multiaccess and broadcast communications," *IEEE International Symposium on Information Theory*, p. 112, June 2004.
- [16] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, pp. 89–103, Jan 2005.
- [17] L. Georgiadis, M. Neely, and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*. Foundations and Trends in Networking, 2006, vol. 1.
- [18] W. Szpankowski, "Stability conditions for some multiqueue distributed systems: Buffered random access systems," *Advances in Applied Probability*, vol. 26, p. 498515, 1994.
- [19] S. Sanghavi, D. Shah, and A. Willsky, "Message-passing for maximum weight independent set," *Proceedings of NIPS*, 2007.
- [20] L. Jiang and J. Walrand, "A distributed csma algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 960–972, Jun. 2010.
- [21] D. Shah and J. Shin, "Randomized scheduling algorithms for queueing networks," *accepted to appear in Annals of Applied Probability*, 2011.